# SQLB: A Query Allocation Framework for Autonomous Consumers and Providers*

J.-A. Quiané-Ruiz†
Atlas group, INRIA and LINA
Université de Nantes
quiane@univ-nantes.fr

Philippe Lamarre
Atlas group, LINA
Université de Nantes
lamarre@univ-nantes.fr

Patrick Valduriez
Atlas group, INRIA and LINA
Université de Nantes
Patrick.Valduriez@inria.fr

## ABSTRACT

In large-scale distributed information systems, where participants are autonomous and have special interests for some queries, query allocation is a challenge. Much work in this context has focused on distributing queries among providers in a way that maximizes overall performance (typically throughput and response time). However, preserving the participants' interests is also important. In this paper, we make two main contributions. First, we provide a model to define participants' perception of the system w.r.t. their interests and propose metrics to evaluate the quality of query allocation methods. This model facilitates the design and evaluation of new query allocation methods that take into account the participants' interests. Second, we propose a framework for query allocation called *Satisfaction-based Query Load Balancing* (*SQLB*). To be fair, *SQLB* dynamically trades consumers' interests for providers' interests. And it continuously adapts to changes in participants' interests and to the *workload*. We implemented *SQLB* and compared it, through experimentation, to two important baseline query allocation methods, namely *Capacity based* and *Mariposa-like*. The results demonstrate that *SQLB* yields high efficiency while satisfying the participants' interests and significantly outperforms the baseline methods.

## 1. INTRODUCTION

We consider distributed information systems with a mediator that allows consumers to access information providers through queries [4, 19]. Consumers and providers are autonomous in the sense that they are free to leave the mediator at any time and do not depend on anyone to do so. For clarity, we henceforth refer to both consumers and providers together as participants. Leaving the mediator is equivalent to depart from the system, but it could be that a participant registers to another competing mediator. Providers can be heterogeneous in terms of capacity and data. Heterogeneous capacity means that some providers are more powerful than others and can treat more queries per time unit. Data heterogeneity means that providers provide different data and thus produce different results for a same query. Providers declare their *capabilities* for performing queries to the mediator. Then, the main function of the mediator is to allocate each incoming query to the providers that can satisfy it. Much work in this context has focused on distributing the query load among the providers in a way that maximizes overall performance (typically throughput and response time) [8, 13, 18, 21], i.e. *query load balancing* (*QLB*). Nevertheless, participants usually have certain expectations w.r.t. the mediator, which are not only performance-related (see Example 1). Such expectations mainly reflect their *preferences* to allocate and perform queries, respectively. Consumers' *preferences* may represent, for example, their interests towards providers (e.g. reputation), preferred providers, or quality of service. Providers' *preferences* may represent e.g. their topics of interests, relationships, or strategies.

EXAMPLE 1. *Consider a provider that represents a courier company. During promotion of its new international shipping service, the provider is more interested in treating queries related to international shipments rather than national ones. Once the advertising campaign is over, the provider's preferences may change. Similarly, consumers expect the system to provide them with information that best fits their preferences.*

In such distributed information systems, query allocation is a challenge. Participants' *autonomy* is the main source of the problem, because they may leave the system if they are too dissatisfied. Thus, it is important to apply a query allocation strategy that balances queries such that participants are satisfied. In this context, the consumer or provider's *satisfaction* means that a query allocation method meets its expectations. To make this possible, the participants' *preferences* must be taken into consideration when balancing queries. However, *preferences* are usually considered as private data by participants (e.g. in an e-commerce scenario, enterprises do not reveal their business strategies). In addition, *preferences* are quite static data, i.e. long-term, while the desire of a provider (resp. a consumer) to perform (allocate) a query may depend on the context and thus is more dynamic, i.e. short-term. For instance, a provider may prefer to perform some kind of queries, but, at some time, it may not desire to perform such queries

*Work partially funded by ARA "Massive Data" of the French ministry of research (Respire project) and the European Strep Grid4All project.
†This author is supported by the Mexican National Council for Science and Technology (CONACyT).

.

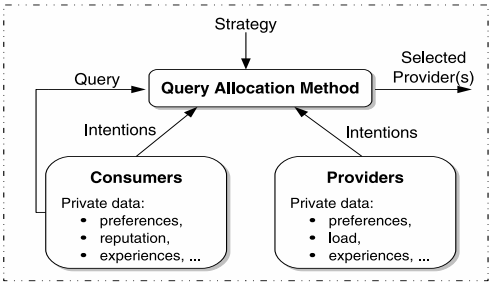



**Figure 1: The Query Allocation Schema.**

because local reasons, e.g. by *overload*. Thus, consumers and providers are required to express their desire to allocate and perform queries, respectively, via an *intention* notion, which may stem e.g. from combining their *preferences* and other private local consideration such as *load* (see Figure 1). When providers can express their *intentions*, queries may not be treated because no provider wants to perform them (as in several economic models [5, 6, 22]). This may hurt the providers as well, because consumers that do not get results from a mediator may simply leave it.

To the best of our knowledge, this problem has not been addressed completely before. Thus, there is no query allocation method that considers any notion of *satisfaction* nor the *intentions* of participants. Our first objective is to propose a model that provides a *satisfaction* notion to characterize how well the mediator meets the participants' expectations in the long-run. Our second objective is to propose a query allocation framework that considers the *satisfaction* and *intentions* of participants.

## 1.1 Motivations

As a motivating example, consider a public e-marketplace where thousands of companies can share information and do business (such as ebay-business [1] and freightquote [2]). Here, business is understood in a very general sense, not necessarily involving money. Each site, which represents a company, preserves its *preferences* for allocating and performing queries. In the rest of this paper, we will often base our concepts definitions on this example. To scale up and be attractive over time, an e-marketplace should *(i)* protect, in the long-run, the participants' *intentions* for doing business, *(ii)* allow consumers to quickly obtain results, and *(iii)* allocate queries so that providers should have the same possibilities for doing business (i.e. to avoid *starvation*) [7].

Consider a simple scenario where a company ($eWine$), which desires to ship wine from France to USA, requests the mediator for companies providing international shipping services, such as freightquote [2]. Here, a query is a call for proposals that providers have to answer so as to provide their services. Consider a second scenario where a company desires to run a specific application, so it requests the mediator for companies providing computing resources (e.g. *CPU* units), as in [3]. The following details are symmetrical for both scenarios. Suppose that $eWine$, to make its final choice, desires to receive proposals from the two best providers that meet its *intentions*, i.e. its expectations. Similarly, providers desire to participate only in those negotiations that involve queries meeting their *intentions*.

In these two scenarios, the mediator must perform several tasks. First, it needs to identify the sites that are able to deal with $eWine$'s query (i.e. to find the providers). Next, the mediator should obtain $eWine$'s *intentions* to deal with such providers and the providers' *intention* to deal with $eWine$'s query[1]. Assume that the resulting list contains, for simplicity, only 5 providers: $p_1, \ldots, p_5$. Table 1 shows these providers with their *intention* to perform the query and $eWine$'s *intention* to deal with each of them. To better illustrate the query allocation problem in these environments, we also show in Table 1 the providers' *available capacity*. However, it is not always possible to know this information since providers may consider it as private. Suppose, then, that $p_5$ is *overloaded*, i.e. has no more resources for doing business, and that $p_2$ and $p_4$ do not intend to deal with $eWine$'s query (notice that this does not means they can refuse it) because e.g. $p_2$ is more interested in its new shipping service to the Asian continent and $p_3$ has bad experience with $eWine$. Also, assume that $eWine$ does not intend to deal with $p_1$ nor $p_3$ since it does not trust them.

**Table 1: Providers for $eWine$'s query.**

| Providers | Prov.'s Int. | Cons.'s Int. | Avail. Cap. |
|---|---|---|---|
| $p_1$ | Yes | No | 0.85 |
| $p_2$ | No | Yes | 0.57 |
| $p_3$ | Yes | No | 0.22 |
| $p_4$ | No | Yes | 0.15 |
| $p_5$ | Yes | Yes | 0 |

Finally, the mediator needs to select the two most available providers, such that $eWine$'s and providers' *intentions* be respected. To the best of our knowledge, no existing e-marketplace is able to do so. In fact, current *QLB* methods (whose aim is to select the most available providers) also fail in such scenarios since neither $p_2$ intends to deal with the query nor $p_1$ is of $eWine$'s interest. Allocating the query to these providers may cause the departure from the system of $p_2$ and $eWine$. The only satisfactory option (regarding the consumer and providers' *intention*) is $p_5$, but allocating the query to it may considerably hurt response time and cause $eWine$'s and $p_5$'s departure from the system. Besides, $eWine$ desires to receive two different proposals.

So, *what should the mediator do in the above scenarios? Should it consider the consumer's intention? the providers' intention? or the providers' available capacity?* In this paper, we address this question so that a query allocation method can decide online what to do according to the status of participants (see Section 5).

## 1.2 Contributions and Organization

The rest of this paper is organized as follows. After defining the problem in Section 2, we present the main contributions of this paper:

- We propose a new model to characterize the participants' expectations in the long-run, which allows evaluating a system from a *satisfaction* point of view. This model facilitates the design and evaluation of query allocation methods for environments with autonomous participants (Section 3). We define the properties to evaluate the quality of *QLB* methods and propose metrics to do so (Section 4).

[1]For simplicity, we assume in this example that *intentions*' values are binary.

- We propose *Satisfaction-based Query Load Balancing* ($SQLB$), a flexible framework with *self-adapting* algorithms for balancing queries while considering participants' *intentions*. $SQLB$ allows to trade consumers' *intentions* for providers' *intentions*. Furthermore, it affords consumers the flexibility to trade their *preferences* for the providers' *reputation* and providers the flexibility to trade their *preferences* for their *utilization* (Section 5).
- We demonstrate, through experimental validation, that $SQLB$ significantly outperforms baseline query allocation methods (namely *Capacity based* and *Mariposa-like*) and yields significant performance benefits. We also show that applying the proposed metrics over the provided model allows the prediction of possible departures of participants (Section 6).

Finally, we survey related work in Section 7 and conclude the paper in Section 8.

## 2. PROBLEM DEFINITION

We consider a system consisting of a mediator $m$, of a set of consumers $C$, and of a set of providers $P$. These sets are not necessary disjoint, an entity may play more than one role. Queries are formulated in a format abstracted as a triple $q = < c, d, n >$ such that $q.c \in C$ is the identifier of the consumer that has issued the query, $q.d$ is the description of the task to be done, and $q.n \in \mathbb{N}^*$ is the number of providers to which the consumer wishes to allocate its query. Parameter $q.d$ is intended to be used within a matchmaking procedure to find the set of providers, denoted by the set $P_q$, that are able to treat $q$. There is a large body of work on matchmaking, see e.g. [11, 14], so we do not focus on this problem and we assume there exists one in the system that is sound and complete: it does not return false positive nor false negatives. We use $N_q$ for denoting $||P_q||$, or simply $N$ when there is no ambiguity on $q$. Consumers send their queries to the mediator $m$ that allocates each incoming query $q$ to $min(q.n, N)$ providers in $P_q$. We only consider the arrival of *feasible queries*, that is, those queries in which there exists at least one provider, which is able to perform them, in the system. For the sake of simplicity we only use, throughout this paper, the "query" term to denote a *feasible query*. Query allocation of some query $q$ among the providers in $P_q$ is a vector $\overrightarrow{All oc}$, or $\overrightarrow{All oc}_q$ if there is an ambiguity on $q$, of length $N$ such that,

$$\forall p \in P_q,\ \overrightarrow{All oc}[p] = \begin{cases} 1 & \text{if p gets the query} \\ 0 & \text{otherwise} \end{cases}$$

As we assume that queries should be treated if possible, this leads to $\sum_{p \in P_q} \overrightarrow{All oc}[p] = min(q.n, N)$. In the following, the set of providers such that $\overrightarrow{All oc}[p] = 1$ is noted $\widehat{P_q}$. Notice that, without any loss of generality, in some cases (e.g. when consumers pay services with real money) the query allocation term just means that providers are selected for participating in a negotiation process with consumers. Providers have a finite *capacity* that may denote e.g. the number of computational units or physical resources they have (depending on their kind of business, e.g. if they provide computational or physical services). Thus, the *utilization* of a provider $p \in P$ at time $t$, $U_t(p)$, denotes how much it is loaded w.r.t. its *capacity*.

A consumer $c \in C$ is free to express its *intention*, denoted by the function $ci_c(q, p)$, for allocating its query $q$ to each provider $p \in P_q$. Results are memorized in the vector $\overrightarrow{CI}_q$. Similarly, a provider $p \in P_q$ is free to express its *intention* for performing a query $q$, denoted by the function $pi_p(q)$. Values of participants' *intention* are in $[-1..1]$. A positive value means that a provider (resp. a consumer) intends to perform (allocate) a query, while a negative value means that a provider (a consumer) does not intend to perform (allocate) a query[2]. A null value, i.e. a 0 value, denotes a participant's indifference. It is up to a participant to compute its own *intentions* by combining different local and external criteria (e.g. utilization, preferences, response time, reputation, past experiences...). The way in which participants compute their *intentions* is considered as private information and not revealed to others.

In these environments, where participants are autonomous, it is crucial that a query allocation method considers their *intentions* in order to preserve the total system capacity, i.e. the aggregate capacity of all providers (e.g. in terms of computational or physical resources). To summarize, we can state the problem as follows.

**Problem Statement.** Given a mediator and autonomous participants, the problem we address is computing and using participants' *intentions* to perform query allocation at the mediator such that response time, system capacity, and participants' *satisfaction* are ensured.

## 3. THE MODEL

We define in this section a model that allows comparing, from a satisfaction point of view, query allocation methods that have different approaches to regulate the system (such as the $QLB$ and economic methods). We are interested in three characteristics of participants that show how they perceive the system. The first one is *adequation*. From a general point of view, two kinds of adequations could be considered: *(i)* the system's adequation to a participant, e.g. a system where a provider cannot find any query it intends to perform is considered inadequate to such a provider, and *(ii)* the participant's adequation to the system, e.g. a consumer issuing queries that no provider intends to treat is considered inadequate to the system. Because of space limitations, we only consider the former in this paper, which we simply call *adequation*. The *adequation* notion helps a participant to evaluate if it might reach its goals in the system. The second one, called *satisfaction*, represents the feeling that a participant has about what it really gets from the system, e.g. a consumer that receives results from the providers it wants to avoid is simply not satisfied. The third one is *allocation satisfaction*, which allows a participant to evaluate the query allocation method regarding its *intentions*. For instance, a provider that performs queries that it does not want is not satisfied with the query allocation method if there exist queries of its interests that it does not get. These last two notions (*satisfaction* and *allocation satisfaction*) may have a deep impact on the system because a participant may decide whether to stay or to leave the system based on them.

Therefore, preserving the participants' *intentions*, in the long-run, is quite important so they stay in the system. A way to achieve this is to make a regular assessment over

[2]It is worth remembering that this does not means it can refuse to perform (resp. allocate) the query.

their $k$ last interactions with the system[3], i.e. the $k$ last query allocations. This is why we define these characteristics over the $k$ last interactions. In addition, those values may evolve with time, but, for the sake of simplicity, we do not introduce time in our notations.

The following definitions may be introduced w.r.t. *intentions* or *preferences* as well (no technical difference). However, it is worth noting that *preferences* are frequently considered as private. In which case, only the participants can apply the following definitions. As far we intend to observe the system's behavior and for simplicity, we just develop the following definitions for *intentions*, which are public.

## 3.1 Consumer Characterization

Intuitively, the consumer's characteristics are useful to answer the following questions: "How well do my expectations correspond to the providers that were able to deal with my last queries?" – *Consumer Adequation* – ; "How far the providers that have dealt with my last queries meet my expectations?" – *Consumer Satisfaction* – ; and "Am I satisfied with the job done by the query allocation process?" – *Consumer Allocation Satisfaction* –. To make this reasoning, a consumer $c$ needs a memory of its $k$ last issued queries, which is denoted by the set $IQ_c^k$.

### 3.1.1 Adequation

This notion characterizes how a consumer considers the mediator. For example, in our motivating example of Section 1.1, $eWine$ considers the mediator as interesting (i.e. adequate), in such a query allocation, because it has providers that $eWine$ considers interesting: $p_2$, $p_4$, and $p_5$. The adequation of a consumer $c \in C$ concerning its query $q$ allocation, noted $\delta_a(c, q)$, is defined as the average of $c$'s shown *intentions* towards the set $P_q$ of providers (Equation 1). Its values are between 0 and 1.

$$\delta_a(c, q) = \Big( \big( \frac{1}{N_q} \sum_{p \in P_q} \overrightarrow{CI}_q[p] \big) + 1 \Big) \Big/ 2 \qquad (1)$$

Let $\overrightarrow{CA}_c[q]$ denote the vector of the adequations obtained by the consumer $c$ concerning its $k$ last queries. Thus, we define the *adequation* of a consumer $c \in C$, $\delta_a(c)$, as the average of the $\overrightarrow{CA}_c[q]$ values (see Definition 1). Its values are between 0 and 1. The closer $\delta_a$'s value from 1, the greater the *adequation* of the mediator to a consumer.

*Definition 1.* Consumer Adequation

$$\delta_a(c) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \overrightarrow{CA}_c[q]$$

### 3.1.2 Satisfaction

This notion helps a consumer to evaluate if the mediator is allocating its queries to the providers it wants. To define the consumer's *satisfaction* over its $k$ last issued queries, we first define the satisfaction of a consumer concerning the allocation of a given query. Intuitively, it corresponds to the average of the *intentions* that a consumer has shown to the providers that have performed its query. Nevertheless, a simple average does not take into account the fact that a consumer may desire different results. Let us illustrate this using our example scenario presented in Section 1.1. Assume that the mediator allocates $eWine$'s query only to $p_2$, to which $eWine$ has an *intention* of 1. In such a query allocation, $eWine$ is completely satisfied (with a satisfaction of 1) even if it did not receive the number of results it desired. Thus, to consider the number of providers desired by a consumer, we define the *satisfaction* of a consumer $c \in C$ concerning the allocation of its query $q$, $\delta_s(c, q)$, as follows,

$$\delta_s(c, q) = \Big( \big( \frac{1}{n} \sum_{p \in \widehat{P_q}} \overrightarrow{CI}_q[p] \big) + 1 \Big) \Big/ 2 \qquad (2)$$

where $n$ stands for $q.n$, i.e. it is the number of results that $c$ desires to obtain. Its values are between 0 and 1.

Let $\overrightarrow{CS}_c[q]$ denote the vector of the obtained *satisfaction* by a consumer $c$ concerning its $k$ last queries. Then, the *satisfaction* of a consumer $c \in C$, $\delta_s(c)$, is a simple average of the $\overrightarrow{CS}_c[q]$ values (Definition 2). Its values are between 0 and 1. The closer the consumer's *satisfaction* value from 1, the more a consumer is satisfied.

*Definition 2.* Consumer Satisfaction

$$\delta_s(c) = \frac{1}{||IQ_c^k||} \sum_{q \in IQ_c^k} \overrightarrow{CS}_c[q]$$

### 3.1.3 Allocation Satisfaction

Let us introduce this notion by means of our example scenario presented in Section 1.1. Assume that $eWine$ has an *intention* of 1, 0.9, and 0.7 for allocating its query to $p_2$, $p_4$, and $p_5$, respectively. Now, suppose that the mediator allocates the query to $p_4$. Such a query allocation corresponds to $eWine$'s high intentions, so $eWine$ is satisfied. However, there is still a provider to which its *intention* is higher ($p_2$). The *Consumer Allocation Satisfaction* notion, $\delta_{as}(c)$ in Definition 3, reflects how much a query allocation method strives to give the best providers to a consumer. Values of the function $\delta_{as}(c)$ are between 0 and $\infty$.

*Definition 3.* Consumer Allocation Satisfaction

$$\delta_{as}(c) = \frac{\delta_s(c)}{\delta_a(c)}$$

If the *allocation satisfaction* of a consumer $c$ is greater than 1, it means that the query allocation method works well for $c$ (from $c$'s point of view). If the value is smaller than 1, the closer it is to zero, the more $c$ is dissatisfied with the query allocation method. Finally, a value equal to 1 means that the query allocation method is neutral for $c$.

## 3.2 Provider Characterization

Intuitively, the provider's characteristics answer the following questions: "How well do my expectations correspond to the last queries that have been proposed to me?" – *Provider Adequation* – ; "How well the last queries I have treated meet my expectations?" – *Provider Satisfaction* – ; and "Am I satisfied with the job done by the query allocation process?" – *Provider Allocation Satisfaction* –. To define these characteristics, a provider $p$ tracks its shown *intentions* for performing the $k$ last proposed queries (allocated to it or not) in the vector $\overrightarrow{PPI}_p$. The $k$ last proposed queries to a provider $p$ is denoted by the set $PQ_p^k$.

[3] $k$'s value may be different for each participant depending on its storage capacity, or strategy. For simplicity, we have assumed here that they all use the same $k$.

### 3.2.1 Adequation

The *adequation* notion helps a provider to evaluate if the queries that consumers issue correspond to its expectations. Considering our example scenario of Section 1.1, one can consider the mediator as adequate to $p_1$, $p_3$, and $p_5$, because $eWine$'s query is of their interest. However, we cannot consider the mediator as inadequate to $p_2$ and $p_4$ only by this query proposition. What is more important for a provider is that consumers generally issue queries of its (the provider's) interests. Thus, we define the *adequation* of a provider $p \in P$, $\delta_a(p)$, as the average of its shown *intentions* towards the set $PQ_p^k$ (Definition 4). Its values are between 0 and 1. The closer the $\delta_a(p)$ value from 1, the greater the *adequation* of the mediator to a provider.

*Definition 4.* Provider Adequation

$$\delta_a(p) = \begin{cases} \Big( \big( \frac{1}{||PQ_p^k||} \sum_{q \in PQ_p^k} \overrightarrow{PPI}_p[q] \big) + 1 \Big) \big/ 2 & if\ PQ_p^k \neq \emptyset \\ 0 & otherwise \end{cases}$$

### 3.2.2 Satisfaction

Conversely to *adequation*, the *satisfaction* notion depends only on the queries that a provider performs. Thus, it helps a provider to evaluate whether it performs queries that allow it to fulfill its objectives or not. To illustrate this notion, suppose that in our motivating example (see Section 1.1) the mediator allocates $eWine$' query to $p_2$. In such a query allocation, $p_2$ is not satisfied since it did not intend to perform the query. Nonetheless, what is more important for a provider is to be globally satisfied with the queries it performs, even if it sometimes performs queries that are not of its interest. Let $SQ_p^k \subseteq PQ_p^k$ denote the set of queries that a provider $p$ performed among the set $PQ_p^k$. Then, the *satisfaction* of a provider $p \in P$, $\delta_s(p)$ in Definition 5, is defined as a simple average of its $SQ_p^k$ values. The $\delta_s(p)$ values are between 0 and 1. The closer the value from 1, the greater the *satisfaction* of a provider.

*Definition 5.* Provider Satisfaction

$$\delta_s(p) = \begin{cases} \Big( \big( \frac{1}{||SQ_p^k||} \sum_{q \in SQ_p^k} \overrightarrow{PPI}_p[q] \big) + 1 \Big) \big/ 2 & if\ SQ_p^k \neq \emptyset \\ 0 & otherwise \end{cases}$$

### 3.2.3 Allocation Satisfaction

As for consumers, a provider is not satisfied when it does not get what it expects. There are different reasons for this. First, it may be because the system does not have interesting resources, i.e. the provider has low *adequation*. Second, the query allocation method may go against the provider's *intention*. This is measured by the *allocation satisfaction* notion. In other words, by means of this notion a provider can evaluate how well the query allocation method works for it. We formally define the *Allocation Satisfaction* of a provider $p \in P$, $\delta_{as}(p)$, as the ratio of its *satisfaction* to its *adequation* (see Definition 6). Its values are in $[0..\infty]$.

*Definition 6.* Provider Allocation Satisfaction

$$\delta_{as}(p) = \frac{\delta_s(p)}{\delta_a(p)}$$

If the *allocation satisfaction* of a provider $p$ is greater than 1, it means that the query allocation method works well for $p$ (from the point of view of $p$). If the value is smaller than 1, the closer it is to zero, the more $p$ is dissatisfied with the query allocation method. Finally, a value equal to 1 means that the query allocation method is neutral for $p$.

## 3.3 Discussion

The proposed model can be applied with three main purposes. First, to evaluate how well a query allocation method satisfies the participants' expectations. Second, to evaluate the reasons of the participants' departures from the system. For example, to know if they are leaving the system because *(i)* they are dissatisfied with the queries they perform, *(ii)* they are dissatisfied with the mediator's job, or *(iii)* the system is inadequate to them. To do so, one has to apply metrics, which reflect a global behavior, over the *adequation*, *satisfaction*, and *allocation satisfaction* of participants (see Section 4). Third, to design new self-adaptable query allocation methods that meet the participants' expectations in the long-run (see Section 5).

Reputation does not directly appear, but it is clear that it has a major role to play in the manner that participants work out their *intentions*. Thus, it is taken into account as much as participants consider it important.

# 4. SYSTEM METRICS

The metrics we use are the same for consumers and providers, and can be used to measure the $\delta_a$, $\delta_s$, $\delta_{as}$, and $U_t$ functions. Thus, for simplicity, the $g$ function denotes one of these four functions and $S$ denotes either a set of consumers or providers, i.e. $S \subseteq C$ or $S \subseteq P$. To better evaluate the quality of a query allocation method for balancing queries, one should reflect *(i)* the effort that a query allocation method does for maximizing or minimizing a set $S$ of $g$ values - *efficiency* -, *(ii)* any change in a set $S$ of $g$ values - *sensitivity* -, and *(iii)* the distance from the minimal value to the maximal one in a set $S$ of $g$ values - *balance* -.

A well-known metric that reflects the *efficiency* of a query allocation method is the *mean* $\mu$ function. Because participants' *characteristics* (see Section 3) are additive values and may take zero values, we utilize the arithmetic mean to obtain this representative number (Equation 3).

$$\mu(g, S) = \frac{1}{||S||} \sum_{s \in S} g(s) \tag{3}$$

However, the *mean* metric might be severely affected by outlier values. Thus, we have to reflect the $g$ values' fluctuations in $S$, i.e. the *sensitivity* of a query allocation method. In other words, we evaluate how *fair* a query allocation method is w.r.t. a set $S$ of $g$ values. An appropriate metric to do so is the *fairness index* $f$ proposed in [9] (defined in Equation 4). Its values are between 0 and 1.

$$f(g, S) = \frac{\big( \sum_{s \in S} g(s) \big)^2}{||S|| \big( \sum_{s \in S} g(s)^2 \big)} \tag{4}$$

Intuitively, the greater the *fairness* value of a set $S$ of $g$ values, the fairer the query allocation process w.r.t. such values. To illustrate the *sensitivity* property, suppose that there exist two competitive mediators $m$ and $m'$ in the motivating example of Section 1.1. Assume, then, that the set of providers registered to $m$ and $m'$ are $P = \{p_1, p_2, p_3\}$

and $P' = \{p'_1, p'_2, p'_3\}$, respectively. Now, consider that the *satisfaction* of such providers are $\delta_s(p_1) = 0.2$, $\delta_s(p_2) = 1$, $\delta_s(p_3) = 0.6$, $\delta_s(p'_1) = 1$, $\delta_s(p'_2) = 0.7$, and $\delta_s(p'_3) = 0.9$. Reflecting the *sensitivity* of both mediators w.r.t. *satisfaction* (0.77 and 0.97 for $m$ and $m'$ respectively), we can observe that enterprises have almost the same chances of doing business in $m'$ than in $m$.

Finally, a traditional metric that reflects the ensured *balance* by a query allocation method is the *Min-Max* ratio. The *Min-Max* ratio $\sigma$ is defined in Equation 5, where $c_0 > 0$ is some pre-fixed constant. Values of the function $\sigma$ are between 0 and 1. The greater the *balance* value of a set $S$ of $g$ values, the better the *balance* of such values. The *Min-Max* ratio is useful to know whether there exists a punished entity $s \in S$, and then, one can evaluate if this is because of the query allocation method or the entity's *adequation*.

$$\sigma(g, S) = \frac{\min_{s \in S} g(s) + c_0}{\max_{s' \in S} g(s') + c_0} \quad (5)$$

These metrics are complementary to evaluate the global behavior of the system, and the use of only one of them may cause the loss of some important information.

# 5. THE SQLB FRAMEWORK

We now present $SQLB$, a flexible framework for balancing queries in considering the participants' *intentions*. A salient feature of $SQLB$ is that it affords consumers the flexibility to trade their *preferences* for the providers' *reputation* (Section 5.1) and providers the flexibility to trade their *preferences* for their *utilization* (Section 5.2). Then, $SQLB$ allows to trade consumers' *intentions* for providers' *intentions* in according to their *satisfaction* (Section 5.3). In this way, $SQLB$ continuously adapts to changes in participants' expectations and *workload*. So far, we assumed that a matchmaking technique has found the set of providers that are able to deal with a query, named $P_q$. Therefore, we only focus on the allocation of $q$ among the $P_q$ set (Section 5.4). Without any loss of generality, participants may differently obtain their *intentions*.

## 5.1 Consumer Intentions

The idea is that a consumer makes a balance between its *preferences* for allocating queries and the providers' *reputation*, in accordance to its past experiences with providers. For example, if a consumer does not have any past experience with a provider $p$, it pays more attention to the *reputation* of $p$. We formally define the *intention* of a consumer $c \in C$ to allocate its query $q$ to a given provider $p \in P_q$ as in Definition 7. Function $prf_c(q, p) \in [-1..1]$ gives $c$'s *preference* for allocating $q$ to $p$, and function $rep(p) \in [-1..1]$ gives the reputation of $p$.

*Definition 7.* Consumer's Intention

$$ci_c(q,p) = \begin{cases} prf_c(q,p)^{\upsilon} \times rep(p)^{1-\upsilon} & if\ prf_c(q,p) > 0 \wedge \\ & \wedge rep(p) > 0 \\ -\Big(\big(1 - prf_c(q,p) + \epsilon\big)^{\upsilon} \times \big(1 - rep(p) + \epsilon\big)^{1-\upsilon}\Big) & else \end{cases}$$

Parameter $\epsilon > 0$, usually set to 1, prevents the consumer's *intention* from taking zero values when the consumer's *preference* or provider's *reputation* values are equal to 1. Parameter $\upsilon \in [0..1]$ ensures a balance between the consumer's *preferences* and the providers' *reputation*. In particular, if $\upsilon = 1$ (resp. 0) the consumer only takes into account its *preferences* (the provider's *reputation*) to allocate its query. So, if a consumer has enough experiences with a given provider $p$, it sets $\upsilon > 0.5$, or else it sets $\upsilon < 0.5$. When $\upsilon = 0.5$ means that a consumer gives the same importance to its *preferences* and the provider's *reputation*.

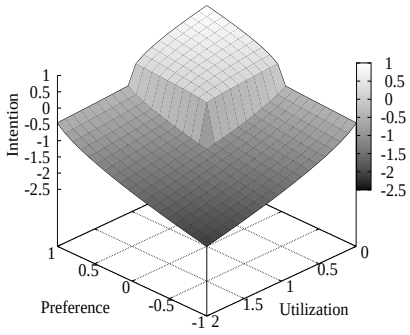



**Figure 2: Tradeoff between *preference* and *utilization* for providers' *intention* when satisfaction is 0.5.**

## 5.2 Provider Intentions

The provider's *intention* is based on its *preferences* for performing queries and its *utilization*. The question that arises is: *what is more important for a provider, its preferences or its utilization?* The importance of the provider's *preferences* and its *utilization* should be balanced on the fly according to *satisfaction*. Intuitively, on the one hand, if a provider is satisfied, it can then accept sometimes queries it does not want. On the other hand, if a provider is dissatisfied, it does not pay so much consideration to its *utilization* and focuses on its *preferences* in order to obtain desired queries. To do so, the *satisfaction* it uses to make the balance has to be based on its *preferences* and not on its *intentions* as defined in Section 3.2. This is possible since a provider has access to its private information. So, we define the *intention* of a provider $p \in P_q$ to deal with a given query $q$ as in Definition 8. The $prf_p(q) \in [-1..1]$ function gives $p$'s *preference* for performing $q$.

*Definition 8.* Provider's Intention

$$pi_p(q) = \begin{cases} (prf_p(q))^{1-\delta_s(p)}(1 - U_t(p))^{\delta_s(p)}, & if\ prf_p(q) > 0 \wedge \\ & \wedge U_t(p) < 1 \\ -\Big(\big(1 - prf_p(q) + \epsilon\big)^{1-\delta_s(p)} \times \big(U_t(p) + \epsilon\big)^{\delta_s(p)}\Big) & else \end{cases}$$

Parameter $\epsilon > 0$, usually set to 1, prevents the *intention* of a provider from taking 0 values when its *preference* is equal to 1 whatever its *utilization* is. Figure 2 illustrates the behavior that the $pi_p(q)$ function takes when the *satisfaction* is 0.5. We can observe that the providers' *preferences* and *utilization* have the same importance for providers. Also, we observe that providers show positive *intentions* to deal with queries only when they are not *overutilized* and want to perform the queries. This helps to keep good response times in the system.

## 5.3 Scoring and Ranking Providers

Given a query $q$, a provider is scored by considering its *intention* for performing $q$ and $q.c$ consumer's *intention* for

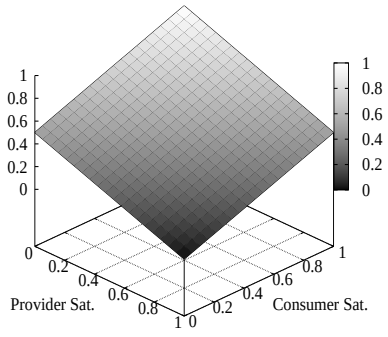



**Figure 3: The values that $\omega$ can take.**

allocating $q$ to it. Considering the mediation process proposed in [10], the score of a provider $p \in P_q$ regarding a given query $q$ is defined as the balance between the $q.c$'s and $p$'s *intentions* (Definition 9).

*Definition 9.* Provider's Score

$$scr_q(p) = \begin{cases} \left(\overrightarrow{PI}_q[p]\right)^{\omega} \left(\overrightarrow{CI}_q[p]\right)^{1-\omega} & if\ \overrightarrow{PI}_q[p] > 0 \wedge \\ & \wedge\ \overrightarrow{CI}_q[p] > 0 \\ -\Big(\left(1 - \overrightarrow{PI}_q[p] + \epsilon\right)^{\omega} \left(1 - \overrightarrow{CI}_q[p] + \epsilon\right)^{1-\omega}\Big) & else \end{cases}$$

Vector $\overrightarrow{PI}_q[p]$ denotes the $P_q$'s *intentions* to perform $q$. Parameter $\epsilon > 0$, usually set to 1, prevents the provider's *score* from taking 0 values when the consumer or provider's *intention* is equal to 1. Parameter $\omega \in [0..1]$ ensures a balance between the consumer's *intention* for allocating its query and the provider's *intention* for performing such a query. In other words, it reflects the importance that the query allocation method gives to the consumer and providers' *intentions*. To guarantee equity at all levels, such a balance should be done in accordance to the consumer and providers' *satisfaction*. That is, if the consumer is more satisfied than the provider, then the query allocation method should pay more attention to the provider's *intentions*. Thus, we compute the $\omega$ value as in Equation 6. Conversely to provider's *intention*, the query allocation module has not access to private information. Thus, the *satisfaction* it uses has to be based on the *intentions*.

$$\omega = \Big(\left(\delta_s(c) - \delta_s(p)\right) + 1\Big) \Big/ 2 \qquad (6)$$

Figure 3 illustrates the tradeoff between the consumer and provider' *intention* for obtaining the $\omega$ value. One can also set $\omega$'s value in according to the kind of application. For instance, if providers are cooperative (i.e. not *selfish*) and the most important is to ensure the quality of results, one can set $\omega = 0$. Finally, providers are ranked from the best to the worst scored, the $\overrightarrow{R}^q$ vector. Intuitively, $\overrightarrow{R}^q[1]$ is the best *scored* provider to deal with $q$, $\overrightarrow{R}^q[2]$ the second, and so on up to $\overrightarrow{R}^q[N]$ which is the worst. As a result, if $q.n <= N$ the $q.n$ best ranked providers are selected, or else all the $N$ providers are selected.

## 5.4 Query Allocation Principle

Algorithm 1 shows the main steps of the query allocation process. Given a query $q$ and a set $P_q$ of providers that are able to perform $q$, the query allocation module first asks for $q.c$'s *intention* for allocating $q$ to each provider $p \in P_q$

**Algorithm 1**: QueryAllocation

```
   Input  : q, P_q
   Output: All⃗oc_q
1  begin
       // Consumer's intentions
2      fork ask for q.c's intentions;
       // Providers' intention
3      foreach p ∈ P_q do
4          fork ask for p's intention w.r.t. q ;
5      waituntil CI⃗_q and PI⃗_q be calculated or timeout;
       // Scoring and ranking providers
6      foreach p ∈ P_q do
7          compute p's score concerning CI⃗_q[p] & PI⃗_q[p] ;
8      rank the set P_q of providers, R⃗^q, regarding scr_p(q) ;
       // Query Allocation
9      for i = 1 to min(n, N_q) do All⃗oc[R⃗^q[i]] ← 1 ;
10     for j = min(n, N_q) + 1 to N do All⃗oc[R⃗^q[j]] ← 0 ;
11 end
```

(line 2 of the Algorithm 1). In parallel, it also asks for $P_q$'s *intention* for performing $q$ (lines 3 and 4). Then, it waits for $q.c$ and $P_q$'s *intentions* or for a given *timeout* (line 5). Once such vectors are computed, as second job, the query allocation module computes the score of each provider $p \in P_q$ by making a balance between the $q.c$ and $p$'s *intentions* (line 6 and 7). Then, it computes $P_q$'s ranking (line 8). Finally, the query allocation module allocates $q$ to the $q.n$ best *scored* providers in $P_q$ and sends the mediation result to the $P_q \backslash \widehat{P_q}$ providers (lines 9 and 10), i.e. to those that were not selected for performing the query. In the case that $q.n < N$, then $q$ is allocated to all $N$ providers. Algorithm 1 can be optimized, but our goal is to show the steps involved in the query allocation process.

# 6. EXPERIMENTAL VALIDATION

Our experimental validation has three main objectives: *(i)* to evaluate how well query allocation methods operate, *(ii)* to analyze if *SQLB* satisfies participants while ensuring good *QLB* because it is not obvious that when adding new criteria, a query allocation method still gives good results for the initial criteria, and *(iii)* to study how well our metrics capture query allocation methods' operation. To do so, we carry out two kinds of evaluations. First, we evaluate the general query allocation process as well as the computed metrics. Second, we evaluate the impact of participants' *autonomy* on performance.

## 6.1 Experimental Setup

We built a Java-based simulator and simulate a *mono-mediator* distributed information system, which follows the mediation system architecture presented in [10]. For all the query allocation methods we tested, the following configuration (Table 2) is the same and the only thing that changes is the way in which each method allocates the queries.

Participants work out their *adequation*, *satisfaction*, and *allocation satisfaction* as presented in Section 3. We initialize them with a satisfaction value of 0.5, which evolves with their last 200 issued queries and 500 queries that have been proposed to them. That is, the size of $k$ is 200 for

**Table 2: Simulation parameters.**

| Parameter | Definition | Value |
|---|---|---|
| nbConsumers | Number of consumers | 200 |
| nbProviders | Number of providers | 400 |
| nbMediators | Number of mediators | 1 |
| qDistribution | Query arrival distribution | Poisson |
| iniSatisfaction | Initial satisfaction | 0.5 |
| conSatSize | $k$ last issued queries | 200 |
| proSatSize | $k$ last treated queries | 500 |
| nbRepeat | Repetition of simulations | 10 |

consumers and 500 for providers. The number of consumers and providers is 200 and 400 respectively, with only one mediator allocating all the incoming queries. We assigned sufficient resources to the mediator so that it does not cause bottlenecks in the system. We assume that consumers and providers compute their *intentions* as defined in Sections 5.1 and 5.2, respectively. For simplicity, we set $\upsilon = 1$, i.e. the consumers' *intentions* denote their *preferences*.

To simulate high heterogeneity of the consumers' *preferences* for allocating their queries to providers, we divide the set of providers into three classes according to the interest of consumers: to those that consumers have *high* interest (60% of providers), *medium* interest (30% of providers), and *low* interest (10% of providers). Consumers randomly obtain their *preferences* between .34 and 1 for *high*-interest providers, between $-.54$ and .34 for *medium*-interest providers, and between $-1$ and $-.54$ for *low*-interest providers. On the other side, to simulate high heterogeneity of the providers' *preferences* towards the incoming queries, we also create three classes of providers: those that have *high* adaptation (35% of providers), *medium* adaptation (60% of providers), and *low* adaptation(5% of providers). The providers randomly obtain their *preferences* between $-.2$ and 1 (*high*-adaptation), between $-.6$ and .6 (*medium*-adaptation) or between $-1$ and .2 (*low*-adaptation). More sophisticated mechanisms for obtaining such *preferences* can be applied (for example using the *TCL* or *Rush* language), but this is beyond the scope of this paper and orthogonal to the problem addressed here. Without any loss of generality, the participants' expectations, in the long run, are static in our simulations. We assume this to evaluate the query allocation methods in a long-term trend, but our model allows expectations to be dynamic.

We set the providers' *capacity* heterogeneity in accordance to the results presented in [20]. We generate around 10% of providers with *low*-capacity, 60% with *medium*, and 30% with *high*. The *high*-capacity providers are 3 times more powerful than *medium*-capacity and still 7 times more powerful than *low*-capacity providers. We generate two classes of queries that consume, respectively, 130 and 150 treatment units at the *high*-capacity providers. *High*-capacity providers perform both classes of queries in almost 1.3 and 1.5 seconds, respectively. We consider in our experiments, without any loss of generality, that providers offer computational services to consumers. Thus, inspired from [8], we assume providers work out their *utilization* as in [16]. We assume that queries arrive to the system in a *Poisson* distribution, as found in dynamic autonomous environments [12]. We do not consider, in this paper, the bandwidth problem and assume that all participants have the same network capacities. Finally, for the sake of simplicity, we assume that consumers only ask for one informational answer (i.e. $q.n = 1$) and all the providers in the system are able to perform all the incoming queries.

## 6.2 Baseline Methods

### 6.2.1 Capacity based method

In distributed information systems, a well known approach is *Capacity based* [13, 18, 21], which allocates each incoming query $q$ to providers that have the highest available capacity (i.e. the least utilized) among the set $P_q$ of providers. *Capacity based* has been shown to operate well in heterogeneous distributed information systems. Hence, we use it as baseline method in our simulations. Note that *Capacity based* does not take into account the consumers nor providers' *intentions*.

### 6.2.2 Economic method

Mariposa [22] is one of the most important approaches to allocate queries in autonomous environments and that has shown good results. Thus, we implemented a *Mariposa-like* method to compare it to our $SQLB$. In this approach, all the incoming queries are processed by a *broker* site that requests providers for *bids*. Providers bid for obtaining queries and then the *broker* selects the set of bids that has an aggregate price and delay under a *bid* curve provided by the consumer. In Mariposa, providers modify their bids with their current load (i.e. $bid \times load$) in order to ensure $QLB$. Note that different economic methods may lead to different performance results than those presented here.

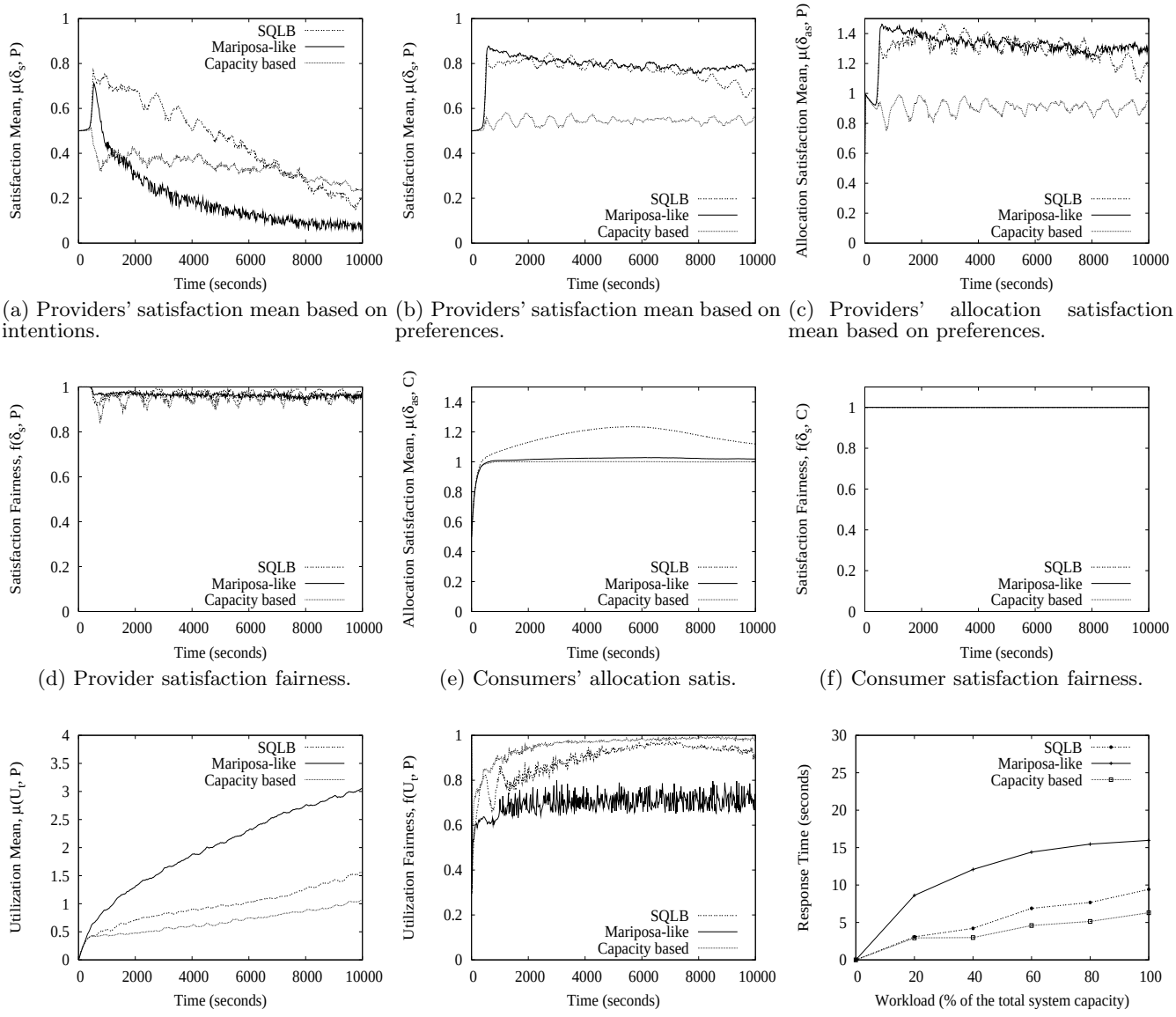
## 6.3 Results

We start, in Section 6.3.1, with an evaluation of the quality of the query allocation methods w.r.t. *satisfaction* and $QLB$. Then, in Section 6.3.2, we evaluate how well these methods deal with the possible participants' departure by *dissatisfaction*, *starvation*, and *overutilization*.

### 6.3.1 Quality results without autonomy

If participants are autonomous, they may leave the system by *dissatisfaction*, *starvation*, or *overutilization*. Nevertheless, the choice of such departure's thresholds is very subjective and may depend on several external factors. Thus, for these first experiments, we consider *captive* participants (i.e. they are not allowed to leave the system). To measure the three methods' quality, we apply the metrics defined in Section 4. However, for space reasons, we can only present two of them. We ran a series of experiments where each one starts with a *workload* of 30% that uniformly increases up to 100% of the total system capacity.

First, we analyze the providers results. Figure 4(a) shows the *satisfaction mean* ensured by the three methods. The *satisfaction* used in this measurement is based on the providers' *intentions*, i.e. what a query allocation method can see. We observe in these results that providers are more satisfied with the $SQLB$ than with the two others. As the *workload* increases, providers' *satisfaction* decreases because their *intentions* decrease as they are loaded (just because *utilization* becomes the most important for them). Thus, $SQLB$ cannot satisfy the providers' *intentions* for high *workloads* since their *adequation* (based on *intentions*) is low. *Capacity based* and *Mariposa-like* do not satisfy the providers' *intentions* from the beginning, simply because




(a) Providers' satisfaction mean based on intentions.

(b) Providers' satisfaction mean based on preferences.

(c) Providers' allocation satisfaction mean based on preferences.

(d) Provider satisfaction fairness.

(e) Consumers' allocation satis.

(f) Consumer satisfaction fairness.

(g) Query load mean.

(h) Query load fairness.

(i) Response times

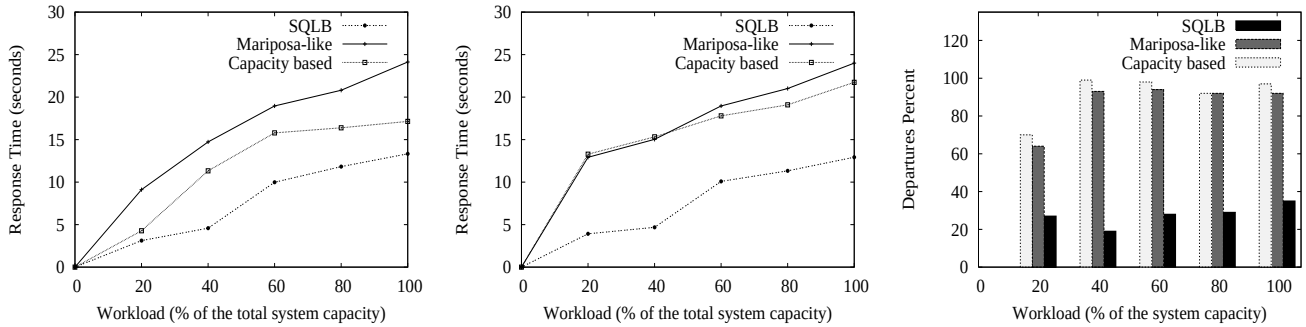
**Figure 4: (a)-(h): quality metrics for a *workload* range from 30 to 100% of the total system capacity when participants are *captive*, and (i): ensured response times with *captive* participants.**

they allocate queries based on other criteria, which do not exactly meet *intention*.

Nonetheless, this does not reflect what providers really feel with respect to their *preferences*. To show this, we need to measure the *mean* ensured by the three methods concerning the providers' *satisfaction* based on their *preferences*. Although we can measure such *satisfaction* in our simulations, this is not always possible since such *preferences* are usually considered as private. Figure 4(b) shows the results of these measurements. We observe that *SQLB* has the same performance as *Mariposa-like* even if it considers the consumers' *intentions*. When the *workload* is close to 100%, the providers' *satisfaction* slightly decreases with *SQLB*. This is because providers pay more attention to their *utilization* for obtaining their *intentions*, thus their *preferences* are less considered by the *SQLB* method.

It is worth noting that, as expected, *Capacity based* is the only one among these three methods that punishes the providers. This is clear in Figure 4(c), which illustrates the ensured *mean* by these three methods with respect to the providers' *allocation satisfaction*. We observe that *Capacity based* severely punishes the providers (*mean* values are always under 1). Then, based on these results, we can predict that when providers will be free to leave the system, *Capacity based* will suffer from serious problems with providers' departures by *dissatisfaction* reasons. Figure 4(d) illustrates the *satisfaction fairness* ensured by the three methods. We see that they guarantee almost the same *satisfaction fairness*. However, as seen in the previous results, this does not mean that providers are satisfied with all three methods.

Now, let us analyze the consumers results. Figure 4(e) illustrates the *allocation satisfaction mean* concerning the




(a) Providers may leave by *dissatisfaction* or *starvation*.

(b) Providers may leave by *dissatisfaction*, *starvation*, or *overutilization*.

(c) Number of providers' departures.

**Figure 5: Impact on performance of providers' departures.**

consumers' *intentions*. We observe that while $SQLB$ is the only one to satisfy consumers, the two others are neutral to consumers (*mean* values equal to 1). These results allows us to predict that *Capacity based* and *Mariposa-like* may suffer from consumer's departures while $SQLB$ does not. The $SQLB$'s *mean* decreases for high *workloads* because of providers. Remember that providers' *satisfaction* decrease because they take care of their *utilization*. So, $SQLB$ pays more attention to providers' *satisfaction* than to the consumers' *satisfaction*. Nonetheless, consumers are never punished. Conversely to providers, the consumers' *satisfaction fairness* has less variations because they are not in direct competition to allocate queries (Figure 4(f)).

Concerning $QLB$, as expected, *Capacity based* better balances the queries among providers than $SQLB$ and *Mariposa-like* (see Figure 4(g)). We can see that the *Mariposa-like* has serious problems to balance queries. It may lose providers by *starvation* or *overutilization* reasons. Figure 4(h) shows that $SQLB$ has some difficulties to be fair (w.r.t. $QLB$) for *workloads* under 40%. In contrast, when the *workload* increases, $SQLB$ pays more attention to $QLB$ and becomes fairer. This demonstrates the high adaptability of $SQLB$ to the variations in the *workloads*.

All above results show that, while *Capacity based* may severely suffer from providers' departures by *dissatisfaction*, *Mariposa-like* may also suffer from providers' departures by *overutilization* problems. Furthermore, these results demonstrate the $SQLB$'s self-adaptability to changes in the participants' *satisfaction* and to the *workload*. This feature makes our proposal highly suitable for autonomous environments.

Finally, Figure 4(i) shows the ensured response times in these environments (with *captive* participants). As is conventional, response time is defined as the elapsed time from the moment that a query $q$ is issued to the moment that $q.c$ receives the response of $q$. As expected, the *Capacity based* method outperforms the two others. However, even if $SQLB$ takes into account the participants' *intentions*, it only degrades performance by a factor of 1.4 in average while *Mariposa-like* does so by a factor of 3.

As concluding remark, we can say that even if not designed for environments where participants are *captive*, $SQLB$ ensures quite good response times and pay attention to the quality of results and queries that consumers and providers get from the system, respectively.

### 6.3.2 Dealing with autonomy

To validate our measurements and intuitions of Section 6.3.1, we also ran several experimental simulations where participants are given the *autonomy* to leave the system. Our main goal, in this section, is to study the reasons by which providers leave the system and evaluate the impact on performance. We evaluate the ensured response times by the three methods in autonomous environments and compare it with those of the *captive* environments (see Figure 4(i)). To do so, we have to set thresholds under, or over, which a consumer or provider decides to leave the system. To avoid any suspicion on the choice of such thresholds, we assume that participants support high degrees of *dissatisfaction*, *starvation*, and *overutilization*. Thus, a consumer leaves the system, by *dissatisfaction*, if its *satisfaction* is smaller than its *adequation*, i.e. the allocation method punishes it. A provider leaves the system *(i)* by *dissatisfaction*, if its *satisfaction* is smaller than its *adequation* minus 0.15, *(ii)* by *starvation*, if its *utilization* is smaller than 20% of its optimal *utilization*, and *(iii)* by *overutilization*, if its *utilization* is greater than 220% of its optimal *utilization*. With a *workload* of 80% of the total system capacity, the optimal *utilization* of a provider is 0.8.

We ran a first series of experiments with different *workloads* where providers are allowed to leave the system only by *dissatisfaction* or *starvation*. The results are shown in Figure 5(a). We observe that $SQLB$ significantly outperforms the others two methods for all *workloads*. Furthermore, we can see that *Capacity based* performs better than *Mariposa-like*. This is because, as seen in Section 6.3.1, the *Mariposa-like* method tends to *overutilize* some providers (those that are the most adapted to the incoming queries), which severely hurts response times.

A second series of experiments allow providers to leave the system by *dissatisfaction*, *starvation*, or *overutilization* (see Figure 5(b)). While $SQLB$ and *Mariposa-like* degrade their performance only by a factor of 1.4 in average (w.r.t. Figure 4(i)), *Capacity based* does it by a factor of 3.5. Figure 5(c) shows the number of provider's departures with the three methods. We observe that, except for a *workload* of 20%, *Capacity based* and *Mariposa-like* lose almost all the providers for all *workloads*. Note that $SQLB$ only loses 28% of providers in average. This demonstrates the high efficiency of $SQLB$ in autonomous environments. We show, in Table 3, an analysis of providers' reasons to leave

**Table 3: Provider's departures reasons for a workload of 80% of the total system capacity.**

| | | *SQLB* | | | | *Capacity based* | | | | *Mariposa-like* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | low | med | high | total | low | med | high | total | low | med | high | total |
| **Dissat.** | Cons. Interest to Prov. | 1% | 5% | 13% | | 5% | 16% | 31% | | 1% | 7% | 11% | |
| | Providers' Adequation | 2% | 9% | 8% | 19% | 3% | 34% | 15% | 52% | 0% | 15% | 4% | 19% |
| | Providers' Capacity | 13% | 6% | 0% | | 13% | 30% | 9% | | 5% | 12% | 2% | |
| | | low | med | high | total | low | med | high | total | low | med | high | total |
| **Starv.** | Cons. Interest to Prov. | 0% | 0% | 4% | | 0% | 0% | 0% | | 0% | 2% | 6% | |
| | Providers' Adequation | 4% | 0% | 0% | 4% | 0% | 0% | 0% | 0% | 3% | 3% | 2% | 8% |
| | Providers' Capacity | 2% | 2% | 0% | | 0% | 0% | 0% | | 3% | 5% | 0% | |
| | | low | med | high | total | low | med | high | total | low | med | high | total |
| **Overuti.** | Cons. Interest to Prov. | 0% | 0% | 6% | | 0% | 0% | 38% | | 0% | 0% | 65% | |
| | Providers' Adequation | 0% | 3% | 3% | 6% | 3% | 8% | 27% | 38% | 1% | 15% | 49% | 65% |
| | Providers' Capacity | 1% | 4% | 1% | | 0% | 18% | 20% | | 0% | 30% | 35% | |

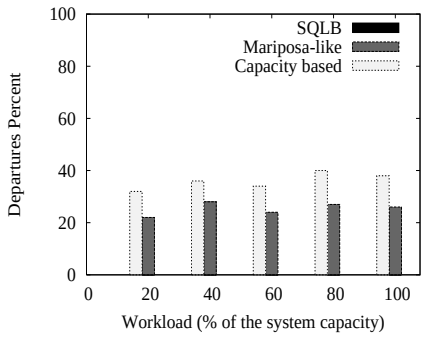



**Figure 6: Consumers' departures.**

the system when the *workload* is 80%. We observe that, as predicted in Section 6.3.1, providers leave the system with *Capacity based* because of *dissatisfaction*, while they do so because of *overutilization* with *Mariposa-like*. Furthermore, the providers that decide to leave in both methods are mainly those that are the most adapted to incoming queries and that consumers desire the most. With *SQLB*, providers leave the system by *dissatisfaction*, but such providers are mainly those that are *low*-capacity. In fact, we can see that *SQLB* mainly maintains the *high*-interest, *high*-adaptation, and *high*-capacity providers in the system.

Finally, Figure 6 shows the consumers' departure by *dissatisfaction* with these three methods. Again, *SQLB* is a clear winner with no consumer's departures. Note that, the consumer's departures have also a direct impact on performance since the less the incoming queries, the less the chances for satisfying providers.

## 7. RELATED WORK

The problem of balancing queries while respecting the participants' *intentions* has not received much attention and is still an open field. In the context of large-scale and heterogeneous distributed information systems, most of the work on query allocation has mainly dealt with the problem of allocating queries to the least utilized providers without any consideration to the consumers or providers' *intentions* [8, 13, 18, 21]. In [16], we proposed a *QLB* method based on the providers' *satisfaction*, but no notion of *satisfaction* nor *intentions* of consumers is considered. In a recent work [17], we provide a set of strategies for balancing queries in distributed information systems with autonomous participants, but that work is complementary to the proposal of this paper and one can use such strategies to improve results (by space reasons, it is not discussed here).

Economic models can claim to take into account the participants' *intentions* and have been shown to provide efficient query allocation in heterogeneous systems [5, 6, 22]. Mariposa [22] is one of the first systems to deal with the query allocation problem in distributed information systems using a *bidding* process. In Mariposa, all the incoming queries are processed by a *broker site* that requests providers for *bids*. Providers bid for acquiring queries based on a local bulletin board. Then, the *broker site* selects a set of *bids* that has an aggregate price and delay under a *bid* curve provided by the consumer. Mariposa ensures a crude form of *load balancing* by modifying the providers' *bid* with the providers' *load*. Nevertheless, our experimentations show that providers suffer from *overutilization*. Besides, queries may not be treated even if providers exist in the system.

In [15], the authors focus on the optimization algorithms for *buying* and *selling* query answers, and the negotiation strategy. Their query trading algorithm runs iteratively, progressively selecting the best execution plan. At each iteration, the buyer sends requests for bids, for a set of queries, and sellers reply with offers (*bids*) for dealing with them. Then, the buyer finds the best possible execution plan based on the offers it received. These actions are iterated until either the found execution plan is not better than the plan found in the previous iteration or the set of queries has not been modified (i.e. there is no new subqueries). This approach uses some kind of bargaining between the buyer and the sellers, but with different queries at each iteration. However, this way of dealing with subqueries optimization is orthogonal to our proposal and one may combine them to improve performances. In [10], the authors propose an economic *flexible mediation* approach that allocates queries by taking into account the providers' *quality* (given by consumers) and the providers' *bids*. In contrast to our approach, the authors inherently assume that participants are *captive*. In addition, their proposed economic model is complementary to our proposal and one can combine them to obtain an economic version of *SQLB*, by computing *bids* w.r.t. *intentions* (which is planned as future work).

Furtheremore, the scope of this paper goes well beyond

related work by characterizing the participants' expectations in the long-run, proposing metrics to analyze them and new algorithms to exploit them.

## 8. CONCLUSION

In this paper, we considered distributed information systems where participants are *autonomous* to leave the system at will. In this context, it is crucial to consider the consumers and providers' *intentions* for allocating and performing queries, respectively, so that their expectations, response times, and system capacity are ensured. We presented a general and complete solution for balancing queries among providers while considering the participants' *intentions*. Our main contributions are the following.

First, we characterized, in the long-run, the participants' expectations in a new model, which allows to evaluate a system from a *satisfaction* point of view. This model facilitates the design and evaluation of new query allocation methods for these environments.

Second, we proposed three different metrics to evaluate the quality of *QLB* methods: *(i)* the *mean* metric that reflects the effort that a query allocation method does for equally either maximizing or minimizing a given set of values, *(ii)* the *fairness* metric that evaluates how fair a query allocation method is, and *(iii)* the *balance* metric that measures the Min-Max values. We proved that using these proposed metrics together, one can predict possible consumer and provider's departures from the system.

Third, we presented the *SQLB* framework for balancing queries in these environments. *SQLB* strongly differs from the related work in several ways: *(i)* it allows providers to trade their *preferences* for their *utilization* while keeping their strategic information private, *(ii)* it affords consumers the flexibility to trade their *preferences* for the providers' *reputation*, *(iii) SQLB* allows trading consumers' *intentions* for providers' *intentions*, and *(iv) SQLB* strives to balance queries at runtime via the participants' *satisfaction*, thus reducing *starvation*.

Finally, we evaluated and compared *SQLB* against two baseline query allocation methods (*Capacity based* and *Mariposa-like*), in two kinds of environments: *captive* and *autonomous*. We showed through experimentation that, by considering together the *QLB* and *satisfaction* of participants, *SQLB* significantly outperforms both. We showed that, unlike the baseline methods, *SQLB* maintains the *high*-interest, *high*-adaptation, and *high*-capacity providers in the system. Moreover, results show that while baseline methods lose more than 20% of consumers (for all *workloads*), *SQLB* has no consumer's departures.

## 9. REFERENCES


[1] The eBay System, http://business.ebay.com.

[2] Freightquote.com, http://www.freightquote.com.

[3] The grid4all Project, http://grid4all.elibel.tm.fr.

[4] R. Miller, editor. *Special Issue on Integration Management. IEEE Data Eng. Bull.*, 25(3), 2002.

[5] D. Ferguson, C. Nikolaou, J. Sairamesh, and Y. Yemini. Economic Models for Allocating Resources in Computer Systems. In S. H. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[6] D. Ferguson, Y. Yemini, and C. Nikolaou. Microeconomic Algorithms for Load Balancing in Distributed computer systems. In *Procs. of the ICDCS Conf.*, 1988.

[7] T. Fong, D. Fowler, and P. Swatman. Success and Failure Factors for Implementing Effective Electronic Markets. *Journal of Electronic Commerce and Business Media*, 8(1):45–47, 1998.

[8] P. Ganesan, M. Bawa, and H. Garcia-Molina. Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems. In *Procs. of the VLDB Conf.*, 2004.

[9] R. K. Jain, D.-H. Chiu, and W. R. Hawe. A Quantitive Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems, DEC-TR-301. Technical report, 1984.

[10] P. Lamarre, S. Cazalens, S. Lemp, and P. Valduriez. A Flexible Mediation Process for Large Distributed Information Systems. In *Procs. of the CoopIS Conf.*, 2004.

[11] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Procs. of the WWW Conf.*, 2003.

[12] E. P. Markatos. Tracing a Large-Scale Peer to Peer System: An Hour in the Life of Gnutella. In *Procs. of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[13] R. Mirchandaney, D. F. Towsley, and J. A. Stankovic. Adaptive Load Sharing in Heterogeneous Distributed Systems. *Journal of Parallel and Distributed Computing*, 9(4):331–346, 1990.

[14] M. H. Nodine, W. Bohrer, and A. H. Ngu. Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. In *Procs. of the ICDE Conf.*, 1999.

[15] F. Pentaris and Y. Ioannidis. Query Optimization in Distributed Networks of Autonomous Database Systems. *ACM TODS*, 31(2):537–583, 2006.

[16] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. Satisfaction Based Query Load Balancing. In *Procs. of the CoopIS Conf.*, 2006.

[17] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. KnBest - A Balanced Request Allocation Method for Distributed Information Systems. In *Procs. of the DASFAA Conf.*, 2007.

[18] E. Rahm and R. Marek. Dynamic Multi-Resource Load Balancing in Parallel Database Systems. In *Procs. of the VLDB Conf.*, 1995.

[19] M. Roth and P. Schwarz. Don't Scrap It! Wrap It! A Wrapper Architecture for Legacy Data Sources. In *Procs. of the VLDB Conf.*, 1997.

[20] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Procs. of the Multimedia Computing and Networking Conf.*, 2002.

[21] N. G. Shivaratri, P. Krueger, and M. Singhal. Load Distributing for Locally Distributed Systems. *Computer*, 25(12):33–44, 1992.

[22] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A Wide-Area Distributed Database System. *VLDB J.*, 5(1):48–63, 1996.